INTRODUCTION

CONCLUSION

INTRODUCTION

Le projet d'administration réseau de ce semestre consistait globalement à fournir au client un architecture réseau convenant aux besoins de l'entreprise. Tout ceci en utilisant des technologies modernes et optimales telles que Docker.

Nous devions fournir un prototype utile reflétant au mieux la solution proposée au client Woody Toys et pouvant permettre au client d'exercer ses différents services de manière efficace et sécurisée.

Ce rapport traite des aspects techniques de la solution de notre groupe.

1. JUSTIFICATION DE L'INFRASTRUCTURE RÉSEAU

a. Schéma Physique

Puisque la DMZ est une zone exposée à internet, nous avons voulu isoler ce sous-réseau avec le réseau local de l'entreprise via un firewall. Celui-ci le sépare du reste du réseau interne et le protège même temps des potentiels attaques provenant de l'extérieur. Les serveurs qui offrent des services externes s'y trouvent : le serveur mail, le serveur Web, le serveur DNS, le serveur SIP…

Pour concevoir cette architecture, nous avons opté pour l'utilisation de deux pares-feux : un séparant la DMZ de l'internet, Et l'autre protégeant le réseau interne de la DMZ ; le but étant d'amener plus de sécurité sur les ressources de l'entreprise aussi petite soit-elle. Si un pirate essaie d'attaquer par exemple la base de données de l'entreprise, il sera confronté à deux obstacles.

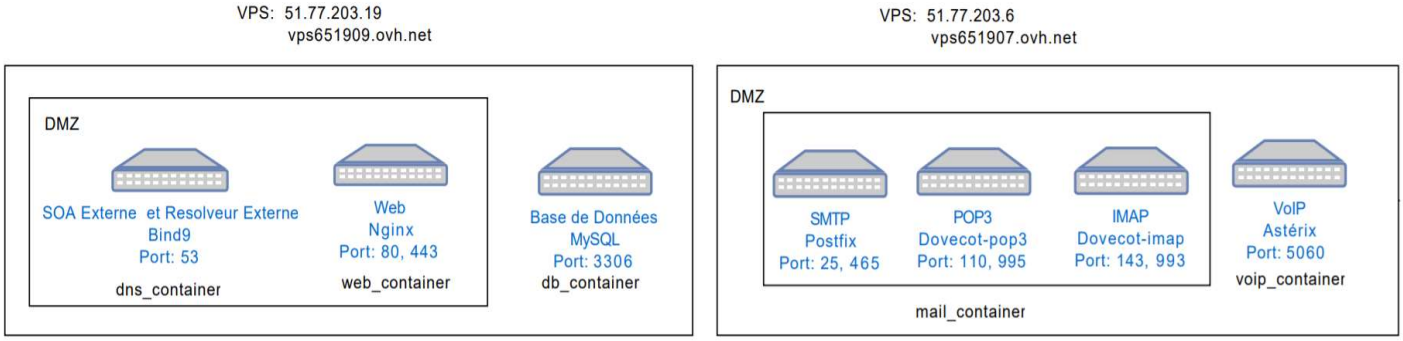
La séparation du réseau du SOA interne et de la base de données du réseau du réseau n'est pas anodine. En effet, le SOA interne contient tout le mappage des hôtes et services locaux et la base de données contient toutes les données confidentielles de l'entreprise. Une potentielle attaque et perte des informations pourrait rendre vulnérable toute l'entreprise. Mettre ces composants avec le reste du réseau local revient à permettre aux utilisateurs internes d'avoir potentiellement accès à ces ressources. Donc une personne mal intentionnée pourrait bien essayer d'attaquer en interne, d'où l'isolation de ces serveurs du Traffic local.

Pour plus d'accessibilité pour les requêtes DHCP, notre serveur DHCP est positionné dans un sous-réseau proche du reste du réseau, de même que le proxy web et le serveur cache. Celui-ci est chargé d'attribuer dynamiquement les adresses IP dans tous les sous-réseaux. Le proxy web a pour fonction d'être l'intermédiaire des utilisateurs internes et internet quand il s'agira de se rendre sur un site et de surfer. Le serveur forwarder interne permet de garder en cache les résolutions DNS des requêtes effectuées par des clients afin de répondre le plus vite possible à des requêtes identiques. Si un utilisateur de l'entreprise demande à se connecter à un site, le serveur cache va regarder en cache s'il a déjà une correspondance la résolution de nom correspondant à l'adresse du site. Si ce n'est pas le cas, il va la demander à la cache externe. Puis il va garder la résolution en cache pendant un temps spécifié par le TTL du domaine. Ainsi, pour la prochaine requête pour le même site, la réponse sera quasi immédiate.

b. Schéma Prototype

Le choix de positionnement des services sur le VPS n'a pas vraiment une justification particulière. Cependant, notre groupe s'est organisé de telle sorte que pour chaque phase, on la réalise sur le VPS du chef de groupe. Ainsi, la phase 1 qui comportait le service DNS, Web et SQL, se trouve sur le VPS 51.77.203.19. Et la phase 2 comportant le service Mail et la phase 3 le service VoIP, se trouvent sur le VPS 51.77.203.6.

Nous avons essayé de faire en sorte que le schéma prototype soit le plus proche possible du schéma physique envisagé pour l'entreprise. Ceci est la raison pour laquelle notre base de données n'est pas accessible les services autres que le service mail. Ce dernier a droit uniquement d'effectuer que des requêtes et saurait insérer, supprimer ou modifier le contenu de la base de données. Les autres serveurs ont accès direct à internet et fonctionnent grâce à la redirection des différents ports du VPS vers les ports des containeurs concernés.



2. ETAT D'AVANCEMENT DES SERVICES MIS EN PLACE

a. Serveur DNS

Pour le service DNS, nous avons mis en place le résolveur externe ainsi que un SOA externe, qui va se charger des requêtes externes. Nous avons aussi géré la configuration des journaux logs pour faciliter le monitoring et gestion du serveur. Et nous avons également le reverse DNS qui est fonctionnel.

b. Serveur Web

Conformément au cahier des charges du projet, nous avons mis en place les trois sites internet dont le site web vitrine **www.wt11.ephec-ti.be** Et aussi les sites b2b.wt11.ephec-ti.be et le site intranet intranet.wt11.ephec-ti.be. Les différents sites sont prêts à l'emploi et ont chacun une page HTML comme page index.

Pour compléter ceux s'il suffit de changer la structure de base dans les fichiers de configurations se situant dans **/var/www/** des différents sites.

c. Serveur de base de données

Pour mettre en place notre base de données, nous avons choisi MySQL. Nous avons installé MySQL et créer une base de données correspondant à l'entreprise WoodyToys où chaque utilisateur sera identifié par un nom d'utilisateur et un mot de passe. Nous avons aussi créé un utilisateur 'admin' associé à l'adresse du VPS lié au service Mail, qui seul y aurait accès.

d. Serveur Mail

La configuration du serveur SMTP n'est pas encore terminé. Nous avons utilisé Postfix pour créer notre MTA. Il fonctionne bien en interne qu'en externe. Nous avons également mis en place Dovecot offrant le service IMAP et POP3. Sa configuration est telle qu'il fonctionne uniquement par SSL.

Spamassassin, le logiciel libre permettant le filtrage des mails considérés comme spams, a été associé avec Postfix et fonctionne très bien. De même, nous avons configuré les règles SPF et DMARC. Par contre, un souci se trouve au niveau du DKIM. Il a bien été configuré mais le démarrage du service opendkim pose des problèmes.

Résultats du test du serveur mail par https://www.mail-tester.com/test-7b232

e. Serveur VoIP

Le serveur de téléphonie IP a été implémenté en utilisant Asterisk suivant le protocole SIP. Nous avons ajouté tous les utilisateurs pour chaque poste dans l'entreprise aves les contextes de l'entreprises. Les appels téléphoniques venant du contexte extérieur sont uniquement acceptés sur le compte public de l'entreprise contact@wt11.ephec-ti.be. Nous avons mis en place un plan assez simple et intuitif pour faciliter la tâche aux employés avec chaque premier chiffre représentant un département précis. Il est aussi possible d'effectuer des appels vidéo.

3. DÉPLOIMENT

a. Serveur DNS

Pour démarrer le serveur VoIP, il faut tout d'abord télécharger l'image

docker pull darrylbilongo/web_image

Pour le lancer il suffit de

Pour mettre en place le service DNS, nous allons tout d'abord importer l'image se trouvant repository DockerHub par la commande :

docker pull manste/projet_admin:dns_image

On va démarrer le container en démon à travers la commande suivante :

docker run -it -d --name dns_container -h ns.wt11.ephec-ti.be -p 53:53/tcp -p 53:53/udp manste/projet_admin:dns_image

Pour se trouver dans le container, on exécute cette commande dans l'environnement /bin/bash :

docker exec -it dns_container bash

Et à l'intérieur de celui-ci, on exécutera la commande « **service bind9 start** » pour démarrer le service.

b. Serveur Web

Pour démarrer le serveur VoIP, il faut tout d'abord télécharger l'image

docker pull darrylbilongo/web_image

Pour le lancer il suffit de

docker run --name web_container -d -p 80:80 -p 443:443 darrylbilongo/voip_image:latest

c. Serveur de base de données

Tout d'abord il va falloir télécharger depuis DockerHub par la commande :

docker pull manste/projet_admin:db_image

Ensuite, il faudra démarrer le containeur par la commande :

docker run -it -d --name db_container --hostname DB -p 3306:3306 manste/db_image bash

Et en se retrouvant dans le containeur par la commande :

docker exec -it dns_container bash

On active le service MySQL ainsi :

service mysql start

d. Serveur Mail

Nous importons l'image depuis le repository sur Dockerhub par la commande :

docker push manste/projet_admin:mail_image

Puis nous démarrons le containeur tout en se rappelant de ne pas oublier de publier les ports :

docker run --name mail_container --hostname mail.wt11.ephec-ti.be --dns 51.77.203.19 --dns-search wt11.ephec-ti.be -it -d -p 25:25 -p 465:465 -p 110:110 -p 143:143 -p 995:955 -p 993:993 -p 23:23 manste/mail_image /bin/bash

Et pour finaliser la mise en place des services, on exécute les commandes suivantes à l'intérieur du containeur :

service mysql start

service dovecot start

service postfix start

service spamassassin start

service postfix reload

service opendkim start

e. Serveur VoIP

Pour démarrer le serveur VoIP, il faut tout d'abord télécharger l'image

docker pull darrylbilongo/voip_image

Puis, il faut lancer le container.

docker run --name voip_container -d -p 5060:5060/tcp -p 5060:5060/udp --hostname sip.w11.ephec-ti.be --dns 51.77.203.19 --dns-search wt11.ephec-ti.be -it darrylbilongo/voip_image:latest

4. OUTILS DE MONITORING ET DÉBOGAGE

a. Serveur DNS

Pour tout ce qui est maintenance, nous avons opté pour la configuration des journaux systèmes. Prenant en compte le fait que les fichiers logs peuvent donc être très volumineux, on a défini le nombre de version su fichier pouvant être mémorisé de deux à trois versions. Ensuite on a mis en place une répartition des fichiers logs en fonction des catégories des requêtes. Ainsi nous avons les fichiers logs pour les requêtes client, pour les atteintes à la sécurité et le service bind9.

Pour le débogage, en plus des fichiers logs qui sont exploitables, nous utilisions par exemple la commande :

named-checkzone wt11.ephec-ti.be /etc/bind/db.wt11.ephec-ti.be

Cette commande vérifie la configuration du fichier de zone. Un autre outil intéressant est « **dig** » qui permettait de vérifier si la résolution de nom du serveur DNS était fonctionnelle. Pour l'observation des ports, l'outil **netstat** avec la commande : « **netstat -nlutp** » permettait de montrer les ports d'écoute TCP et UDP. Dans notre cas, il permettait de vérifier si le port 25 du service DNS était ouvert.

b. Serveur Web

Pour le serveur web, le monitoring consiste essentiellement à surveiller les fichiers logs d'access et d'erreur pour chacun hôtes virtuels. Ceci serait simplifié à l'aide d'un outil de monitoring pour nginx qui est **ngxtop**.

En plus de ces outils de monitoring, pour débugger, il est possible d'analyser le comportement de base des serveurs en utilisant l'outil curl. De plus il est possible d'activer le debuggage sur les fichiers logs d'erreur. Il suffit de rajouter l'option "debug".

curl www.wt11.ephec-ti.be

error_log /var/log/nginx/error.log debug ;

Enfin, il est toujours meilleur d'augmenter le niveau des fichiers logs si les accès aux sites deviennent très récurrents pour une meilleure analyse du trafic.

c. Serveur de base de données

Par défaut, MySQL permet l'enregistrement des différents logs. Ce fichier existe dans le répertoire /var/log/. Jusqu'à présent, il demeure le seul moyen de gérer les erreurs pouvant être liées au service mysql.

La commande « **netstat -nlutp** » a toujours été utilisé pour vérifier le VPS tout comme le containeur écoute sur les ports prévus pour le service.

d. Serveur Mail

Etant donné que la configuration des logs du service postfix et dovecot n'a pas été faite, notre seul moyen d'obtenir les messages d'erreurs en cas de failles de notre configuration était « **docker logs –details mail_container** » avec **mail_container**, le nom du container contenant le service mail.

Un autre outil de débogage est une commande proche de **netstat** de par sa fonctionnalité qui est « **nmap -sS 127.0.0.1** », permettant de scanner les ports et de montrer celle ouvert.

Pour tester la connexion du serveur de base de données avec la configuration, un outil très intéressant est **postmap** de Postfix, qui retourne 1 en cas de réussite et un message d'erreur précisant la ligne où elle se trouve. Par exemple si on désire vérifier que notre serveur SMTP parvient à se connecter, faire une requête SQL vers le serveur de base de données et vérifier l'existence du compte mail : **contact@wt11.ephec-ti.be** , on exécutera la commande :

postmap -q contact@wt11.ephec-ti.be mysql:/etc/postfix/mysql-virtual-mailbox-maps.cf

/etc/postfix/mysql-virtual-mailbox-maps.cf étant le fichier sur lequel se base Postfix pour effectuer la connexion à distance et pour effectuer la requête SQL. **Postmap** permet également la vérification des fichiers de configuration de Postfix. Comme pour le fichier main.conf, la commande suffit pour le faire : **postmap -q main.cf**

e. Serveur VoIP

Pour le monitoring du serveur SIP il serait intéressant de revoir les messages envoyés par Asterisk. De plus, une analyse fine des traces des protocole SIP et RTP utilisant Wireshark serait aussi intéressante aussi bien dans le monitoring que pour débugger le serveur.

5. DIFFICULTÉS RENCONTRÉES

a. Serveur DNS

En plus de l'emploi de temps restreint et la difficulté dans la compréhension du sujet à savoir créer le service DNS avec Docker, la plus difficile étape dans cette partie fut la création de notre premier Dockerfile. De plus que celui-ci se base sur les fichiers externes pour mettre en place la configuration et ensuite déboguer quand rien ne fonctionne.

Une autre difficulté à mettre en place le DNS était aussi lié aux manques de connaissances des différents outils de débogage au début. En plus de cela, une erreur qui empêchait le service DNS de fonctionner en dehors du container était l'ACL mis en place pour la résolution externe.

b. Serveur de base de données

Au cours de l'évolution du projet, nous avions découvert des failles lors de la connexion à distance avec notre serveur MySQL, qui nous a fallu un moment pour débugger. Des erreurs liées à l'utilisateur ayant accès à la base de données, l'insertion du mot de passe utilisateur au cours des l'installation de MySQL dans l'image qu'on a pu débugger en autorisant le VPS 51.77.203.6 à se logger et en trouvant un moyen d'écrire le mot de passe d'utilisateur en clair au cours de l'installation de MySQL. Evidemment ça apporte un risque de sécurité niveau de notre base données : tout le monde peut avoir accès au mot de passe utilisateur.

c. Serveur Web

Nous n'avons pas réussi à mettre en place la sécurisation de notre serveur Web. En effet, nous avions privilégié le fait que les sites soient accessibles dans un premier temps. Cependant la gestion des certificats a été beaucoup plus difficile que prévu en utilisant docker. Une solution était de mettre les différents certificats en clair mais nous ne voulions pas nous aventurer sur ce terrain-là parce que nous subissions déjà une attaque sur le serveur SIP. Ceci bien que nous ayons essayé de le protéger à l'aide de Fail2ban.

d. Serveur Mail

Nous avons également eu des soucis au départ avec la configuration de Dovecot. La création du certificat à l'intérieur de l'image sans passer par le prompt était un défi, car un fois passé par là, on observait une erreur de la part du Dockerfile. Donc il faut trouver un moyen d'en créer sans passer par le prompt de openssl. Moyen finalement trouvé, car maintenant notre serveur IMAPS et POP3 fonctionne uniquement par IMAPS et POP3S. Le débogage de Spamassassin a pris un certain temps pour être réaliser. De plus que la configuration changeait aussi en fonction des versions des linux et surtout de celle d'Ubuntu. Et la documentation qui nous convenait, concernait le plus souvent Centos. Ce qui donna encore un peu plus de labeurs pour adapter le code avec ce qu'on avait déjà réalisé.

Bien que Opendkim fût configuré, il nous affiche hélas une erreur au moment du démarrage du service : **Starting OpenDKIM: install: invalid group**. Son débogage nous a pris beaucoup de temps et malgré cela, rien n'a été trouvé pour résoudre le problème.

e. Serveur VoIP

Nous n'avons pas eu de soucis particuliers concernant la configuration du serveur en lui-même. Cependant nous avons subi une attaque par la force sur le VPS où se trouvait le container en route. Ceci rendait donc difficile la lecture des informations sur le terminal Asterisk désigné justement pour vérifier les configurations et aussi savoir qui s'est enregistré en tant qu'utilisateur sur le serveur SIP. Nous avions donc trouvé des alternatives qui permettent de lire les informations fournisses par les commandes sans devoir être dans le terminal Asterisk.

CONCLUSION

Ce projet fût très instructif dans son ensemble et nous sommes d'accord pour dire qu'il a probablement été le plus intéressant depuis le début de notre formation. L'enthousiasme de se plonger dans les différents types de serveurs que pourrait comporter une entreprise actuellement a été le moteur de notre motivation à s'investir dans le projet. Notre solution nous semble créative et adapté au besoin du client. Avec plus de temps nous aurions fait quelque chose d'encore plus élaboré. Nous regrettons tout de même de ne pas avoir pu nous investir un peu plus dans la sécurisation de nos services sachant que c'est vraiment l'une des choses les plus importantes du rôle d'administrateur réseau.